Sensors and Actuators Integration in Embedded Systems

P. Hara Gopal Mani[1], Member IEEE, Dr. Ibrahim Khan[2], and Dr. KVSVN Raju[3],
[1]Professor, Vignana Bgarathi Institute of Technology, Aushapur, Ghatkesar (M), RR Dist.501301India
e-mail:gopalmaniph@yahoo.com
[2] Director, RGUKT, Nuzvid, Krishna District, AP, India
e-mail: profibkhan@yahoo.co.in
[3] Professor, Computer Science & System Engineering, AUCE (Autonomous), Visakhspatnam, AP, 530003 India
e-mail:kvsvn.raju@gmail.com

Abstract: **Sensors and actuators are critical components of several embedded systems (ES) and can trigger the incidence of catastrophic events [1-3]. Sensor and actuator faults detection is difficult [2, 3] and impacts critically the system performance. While integrating sensors and actuators with the rest of (sub) system, there is a need to identify all failure modes and rectify them. Several researchers have addressed software integration issues [6, 7]; however sensors and actuators integration issues were not addressed so far. This paper focuses on the problem of integration testing of sensors and actuators in ES.**

A fault model, applicable to both sensors and actuators is proposed based on embedded system model in [12] and some of the observed faults are described. They are similar to the control flow and data flow faults [10]. The integration testing of sensor / actuator within ES is the problem of diagnosing faulty machine in a sequence of two CFSM [11]. For solving the diagnostic methodology [13] is used. The integration testing of sensors / actuators is a subset of general diagnostic problem. The sensor/actuator integration method is described with an example and shows that solution exists for the case of integration. Manifestation of faults in real interfaces is described from integration point of view.

Keywords: **Sensor and actuator fault model, manifestation of faults communicating FSM, sensor/ actuator integration in ES, Complex Embedded System.**

I.INTRODUCTION

Sensors and actuators are critical components of an Embedded Systems (ES) in a large number of complex real life applications, for example, Radars, Aircraft, Process Industry and host of other systems that makes use of automatic control. In these Complex Embedded Systems (CES) a fault in the sensors or actuators can trigger the incidence of catastrophic events [1-3]. Sensor and actuator faults detection is difficult [2, 3] and impacts critically the system performance. Currently system development effort is shifting from the design and implementation phases to the system integration and testing phases [4]. In general, Systems Integration phase is underestimated, even in normal projects [5] and as such this phase is a bigger challenge for CES. Several researchers have addressed software integration issues only [6, 7]. Sensors/actuators integration with Subsystem or Embedded Processing Component (EPC) was not addressed adequately so far.

EPC integration testing with sensor or actuator is required to assure that the 'Subsystem under test (SUT)' meets the specifications. While conformance testing is based on building 'relevant' models, fault-based approaches generate faults of required type, and generate test sequence (TS) to expose them. Both conformance testing and composition testing strategies rely on specific fault models [8]. Sensor and actuator faults can lead to safety critical situations [1-3] and as such the integration testing of these essential components assumes great importance in the development of ES.

The focus of this paper is on integration testing of the sensors and actuator with EPC. EPC, Sensor and actuator are modeled as deterministic finite state machines (DFSM) and their possible observed faults (operations) are described. In [9] these possible operations are referred to as type 1 to 4, for modelling possible alterations of the specification machine made during the implementation process.The effect of sensor (actuator) faults is similar to the control flow and data flow faults in the ES [10]. The sensor (actuator) integration with EPC is viewed as faulty machine identification [11] problem within SUT composed of Communicating FSM's. Sensor and actuator behaviour and type of faults are briefly described in section III. EPC and sensor fault models are developed based on embedded system model in [12]. Section IV deals with preliminaries of system of two CFSM and also considers integration testing of sensor and EPC. With an example the diagnostic methodology [13] is described and shown that the integration testing is a subset of faulty component identification problem. Section V discusses manifestation of faults in real interfaces from integration point of view.

II.RELATED WORK

Detection of Sensor and actuator faults is difficult [2, 3] and impacts the system performance critically based on the applications. Sensor faults have been studied in process control [14, 15], Aerospace [3], Wireless sensor networks [2] and other applications [16-18]. A features list for modelling sensor faults and common sensor data faults are given [2]. In fault detection and isolation (FDI) of system faults, faulty sensor outputs will also cause inaccurate diagnostic results or false alarms. Most sensor FDI methods require the determination of explicit state-space or transfer function models [16-18] and some are using Principal Component Analysis (PCA) [16]. FDI methods are based,

DOI: 01.IJNS.02.02.79

e.g., on parameter estimation namely (Extended) Kalman filters, parity equations or state observers. Signal model approaches were developed with an aim to generate several symptoms indicating the difference between nominal and faulty status. Based on different symptoms fault diagnosis procedures follow, determining the fault by applying classification or inference methods [15]. Approaches to FDI for dynamic systems using methods of integrating quantitative and qualitative model information, based upon soft computing (SC) methods are surveyed in [17]. In [1, 3, 15] both sensor and actuator faults are considered. In [1] an actuator and sensor FDI system for small autonomous helicopters is reported. Fault detection is accomplished by evaluating change in the behaviour of the vehicle with respect to the fault-free behaviour, which is estimated by using observers. In [3] efforts to identify and classify critical failure modes for Electro-mechanical actuators (EMA) are described. Also a diagnostic algorithm based on an artificial neural network is reported for EMA.

III.MODEL OF SENSOR

Sensors are always directly in contact with the environment (input measurands) sensing the input energy from the measurands by means of a "sensing element" and then transforming it into another form by a "transduction element." The sensor-transduction elements and associated electronics combination will be referred to as the "Sensor Electronics (SE)" or simply as **sensors**. Measurands relates to the quantity, property, or state that the transducer seeks to translate into an electrical output.

Sensor Faults carry different meaning depending on the application and they can be viewed from data and system point [2]. Analog sensors typically have four types of anomalies: bias, precision degradation, complete failure (dead), and drift. Generally, accurate and reliable sensor readings are essential for over all system performance [2, 16, 17].

Actuators typically accepts a control input (mostly an electrical signal) and produces a change in the physical system by generating force, motion, heat, and so forth. Dead-zone, backlash, and hysteresis are typical nonlinearities found in various actuators. These types of nonlinearities can have adverse effects on control loops. Sensors and actuators also may develop several types of faults and fail in a variety of ways. They may also vary with age, wear, or corrosion.

Sensors and actuators are critical components and are in continuous interaction with the environment. As such their behaviour may be represented as a deterministic finite state machine (DFSM). In the sequel only sensor fault model is considered and the results obtained can be easily extended to actuators. While extending it should be noted that actuators act on output provided by the controller. In the following the *sensor model* is briefly described which formally captures errors in the interface between the "transduction process" and the associated connecting hardware. This model is based on embedded system model proposed in [12], for capturing errors in hardware plus interface and driver software.

As illustrated in figure 1(a) a sensor may be regarded as its "transduction process" functionality / behavior (transfun) encapsulated by mechanical housing and connecting hardware (conhd) (both electrical / electronics and mechanical). Notably all communications between the environment and its transfun pass through the conhd. In fig 3(a) this is shown by letting the inputs from the environment pass through the conhd towards the function via the interaction-point(s) (ip). The ***ip*** are the unlabeled arrows that are considered to be abstract notions. Similarly the functional outputs pass via ***ip*** through the connecting hardware. Each ***ip*** is assumed to be related to exactly one input or output pin-connection. In fig 1(a) this is shown by letting the inputs (a; b; c; d; e) from the environment pass through the connecting hardware via ***ip.*** Likewise the outputs (0; 1; 2; 3) generated have to pass via ***ip*** through the mechanical hardware. Each ***ip*** is assumed to be related to precisely one input or output connection.

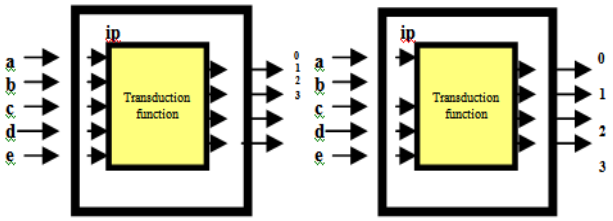

Figure 1(a): Sensor Model (b) Sensor with missing input fault

This model represents sensor behavior faults manifested through ***ip*** only. Generally, during integration testing, individually tested and found correct components are only used. Although three fault cases are possible, only the case given below need to be considered. Case: ***conhd*** not correct and ***transfun*** not correct.

Figure 1(b) shows that the *b-input* is disconnected with the *transfun*. This corresponds to the situation where some "*insignal1"* of the system is not connected such that the *transfun* will never receive the input, and therefore the application of the "*insignal1"* will cause no effect. The fault in fig 1(b) implies that the input *b* never will occur as input to the *transfun*. Referring to the model in fig 1 (b), the possible faults are missing input and output, redirected input and output. These faults are "input variables" of Controller (algorithm) that determines the (next) out put state (combined action of controller and actuator). It can be seen that the effect of sensor faults on ES is similar to that of control and data flow faults [10].

IV.INTEGRATION TESTING OF SENSORS

The objective of integration testing is to uncover errors in the interaction between the components and their environment. This implies testing interfaces between components to assure that they have consistent assumptions and communicate correctly.

A System of sequence of CFSM

Complex embedded systems are being modelled as a sequence of communicating FSMs (SCFSM) of several

DOI: 01.IJNS.02.02.79

FSMs *Ai*, *i*=1, ..., *k*. It is assumed that the component FSM *Ai* is deterministic FSM which communicate asynchronously with each other through bounded input queues, in addition to their communication with the environment through their respective external ports. SE and the ES to be integrated are modeled as a sequence of CFSM (SCFSM) components (Fig.2). Consider the two CFSMs: A1 and A2, where X and Y represent the externally observable input/output actions of A1. Let U and Z alphabets represent the internal (observable) input/output interactions between the two components A1 and A2. Here A1 and A2 are the SE and the mixed hardware-software implementation of EPC (IEPC).

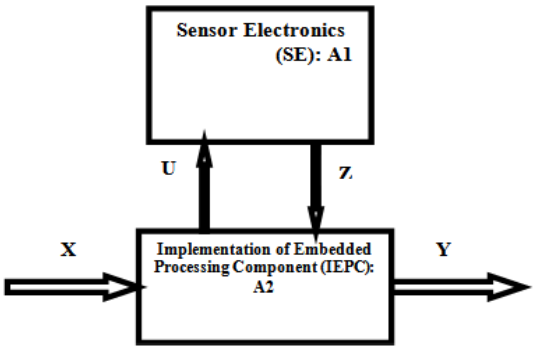

Figure 2: Embedded System Components for Integration

It is assumed that the system has at most one message in transit, i.e. the next external input is submitted to the system only after it produces an external output to the previous input. Then the collective behaviour of the two communicating FSMs can be described by a finite *product machine (PM)* and finite *composed machine (CM)*, since the number of all possible global states of the system is limited. The PM describes the joint behaviour of component machines in terms of all actions within the system, whereas the CM describes the observed behaviour in terms of external inputs X and outputs Y. So, *PM* = A1X A2 and *CM*=A1Ê%A2. We assume that the component machines A1 and A2 of a SCFSM are deterministic and so the system does not fall into a live-lock, then the composed machine *CM* is deterministic [9-11].

Consider for example the two machines A1 and A2 as shown in Fig.3 and their corresponding Reference System *CM*=A1Ê%A2 as shown in Fig 4. The set of external inputs is X={x1, x2, x3}, the set of external outputs is Y = {y1, y2, y3}, the set of internal inputs is U = {u1, u2, u3}, and the set of internal outputs is Z = {z1, z2, z3}.

It is not always possible to locate the faulty component of the given system when faults have been detected in a SUT. Here the two cases are:

1. Only one of the components A1 or A2 is faulty and do not know which one.
2. One of the components A1 or A2 is faulty and the correct one known.

Case 1 is known as diagnostic problem [11, 13, 21]. Here interest is sensor integration, which corresponds to the case 2, where in it is assumed that A2 to be correct and A1 faulty.

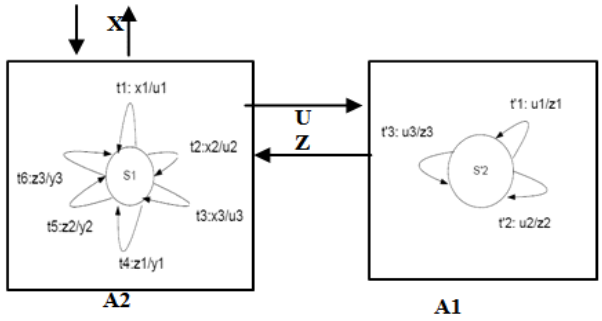

Figure 3: A system of two CFSMs, A1 and A2.

B. A Fault Model for the System of CFSMs

First consider the general problem of case1. The proposed sensor fault model (section 2) can represent faults due to errors developed in sensors as missing (I/O), disconnected and redirected output. Generally the fault model based on output and transfer faults is typically used for diagnosing the system decomposed into components, where only one component may be faulty [20]; and only output and transfer faults of a deterministic FSM are considered here.

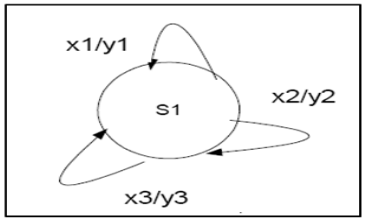

Figure 4: Reference System *CM*=A1Ê%A2

C. The Diagnostic Methodology

Let A1 and A2 be the deterministic FSMs representing the specifications of the components of the given system (SE and ES); while B1 and B2 are their implementations respectively. *Conformance testing* determines whether the I/O behaviour of A1 and A2 conforms to that of B1 and B2, or not. A test sequence that addresses this issue is called a checking sequence. An I/O difference between the specification and implementation can be caused by either an incorrect output (output fault) or an earlier incorrect state transfer (state transfer fault). A standard test strategy is [11]: Homing the machine to a desired initial state, Output Check for the desired output sequence, by applying an input (test) sequence (TS) and Tail State Verification. It is assumed that any test sequence (TS) considered has been generated by using a standard test strategy. The method for diagnostic test derivation is outlined [13] with an example.

D. An Example

In this example, a reset transition **tr** is assumed to be available for both the specification and the implementation. The symbol "r" denote the input for such a transition and the null symbol "-" denote its output. A reset input "r" resets both machines in the system to their initial states. Suppose the test suite TS = {r-x1, r-x2, r-x3 } is given for the two CFSMs specification shown in Figure 3.

DOI: 01.IJNS.02.02.79

Assume that the implementation of A1and A2 is equal to the specification with the exception that t'1 of A1 has the **output fault** z2. The application of TS to the specification of Figure 3 and its corresponding implementation of A1 and A2 generates the expected and observed output sequences given in Table 1. A difference between observed and expected outputs is detected for test cases tc_1. Therefore, the symptom is: Symp1 = $(o_{tc1,1} \neq \hat{o}_{1,1})$.

TABLE I:
EXPECTED AND OBSERVED OUTPUT SEQUENCES

Test suite TS = {r-x1, r-x2, r-x3 }				
S.No	Test Case (tc_i)	tc_1	tc_2	tc_3
1	Inputs	r, x1	r ,x2	r, x3
2	Specified transitions	t1, t'1, t4	t2, t'2, t5	t3, t'3, t6
3	Expected Output	y1	y2	y3
4.	Observed Output	y2	y2	y3

Corresponding to the above symptom, determine the following conflict paths for both machines A1 and A2, which are equal to tentative candidate faulty transitions for this particular example:

ConfpA21 = t1, t4

ConfpA11 = t'1

Corresponding to these tentative candidate transitions, compute the following tentative diagnostic candidates for A1 and A2:

TdiagcA11 = A1 where t'1 has been changed to u1/z2 instead of u1/z1

TdiagcA12 = A1 where t'1 has been changed to u1/z3 instead of u1/z1

TdiagcA21 = A2 where t1 has been changed to x1/u2 instead of x1/u1

TdiagcA22 = A2 where t1 has been changed to x1/u3 instead of x1/u1

TdiagcA23 = A2 where t4 has been changed to z1/y2 instead of z1/y1

TdiagcA24 = A2 where t4 has been changed to z1/y3 instead of z1/y1

Notice that TdiagcA12, TdiagcA22, and TdiagcA24 do not explain all observable outputs of the SUT, and thus are not considered as diagnostic candidates. For example, if the fault is as specified in TdiagcA12 (t1:x1/u3), the SUT should produce the external output y3 for the external input r-x1 of Tc1; however, it produces the external output y2 as shown in Table 1. The remaining tentative diagnostic candidates are considered as diagnostic candidates DiagcA11, DiagcA23, and DiagcA21, respectively. For these candidates, the following composed machines are formed:

DiagcA11 Ê% A2; DiagcA23 Ê% A1; and DiagcA21 Ê% A1.

These machines are equivalent, and therefore faulty machine can not be determined by testing the composed system in the given architecture. If any of the machines are equivalent additional tests described in [21] are generated, for distinguishing between the diagnostic candidates. For sensor integration, because of our assumption, additional tests need not be generated and A1 is faulty.

a. Integration of Actuator

Actuator integration testing in ES is similar to sensors.

b. Discussion

The diagnostic method assumes that a test sequence (TS) is given and has been generated by using a standard test strategy [11]. There are several FSM based test generation [10, 12 and 19] methods are available with varying abilities to detect certain errors of the fault model given in section III for generating TS. However, it is shown that functions can be used as, partial specifications [22]. Sensor calibration is an inevitable requirement, in many applications and is carried out in controlled environment. After calibration a sensor output response (function) for an input pattern (function) is available. Sensor integration testing is normally done after calibration. The sensor input pattern can be taken as TS for integration testing. With the help of calibration information and monitored outputs sensor integration with EPC can be successfully carried out. When faults have been detected in a system under test (SUT) the faulty component of the given system is inferred under the assumption that the other machine is correct. This is required to be ascertained by the tester during integration test. This intern implies that the output responses of the component to be observable for monitoring by the tester. For the case of sensor integration, this implies that sensor output response should be observable for monitoring. This can be achieved by providing a separate 'monitoring port' for sensors and actuators.

V.MANIFESTATION OF FAULTS THROUGH INTERFACES

Sensor and actuator faults affect various levels of system hierarchy. The lowest level is the hardware level, the next level is low level control where sensor information is processed and the top level is high level control which determines the systems behaviour. Clearly sensor or actuator failures affect at the hardware level of the system. The view of CES as a sequence of communicating (processes) FSM allows to define system (level) problems that occur during the design stage into three groups [23, 24]: Component -To-Communication Architecture (COMP2COMM) problems, Component-To- Component (COMP2COMP) problems, and Communication (COMM) problems. But it is sufficient to consider faulty communication channel and assume fault-free processes as it permits ignoring their internal structure completely [25]. When each process is decomposed, new communication links become visible, and based on the "internal" faults in the original process, the newly visible communication links are modelled as faulty. The fault model described in the previous section is a behaviour model and any of the possible type of fault mentioned is manifested through ip only. The detected fault maps in real terms through the above communication link architecture [23, 24] between SE and IEPC. From integration point of view "manifestation

DOI: 01.IJNS.02.02.79

of faults" are observed via I/O or bus connections only. That the embedded system behaviour is to be related by the set of input vs. set of sequence of observed outputs. Hardware/ Software Interface links the software part and the hardware part in the system. It can comprise any, all or some of the following interfaces shown in figure 8.

In fig 8(a, b) two asynchronous interfaces, an input interface (Tri-state-gate) and in-out interface, respectively are shown. The input is 'x', output is 'y' and 'g' is the control. The in-out interface shown in figure 8(b) can work as either input interface or output interface (but not both) at any time. The two synchronous interfaces shown in fig 8(c, d) are used in support of synchronous communication between the software component and the hardware component of ES. The software component of the synchronous interface inputs data from the hardware component fig8(c), and RD is a control signal for reading. Here, the software program waits for the flag '*F*' to be set to indicate availability of an input data on the data Bus. The *reset* wire of the flag is linked to the control wire ", which is driven by the control signal RD. In fig8(d) software component of the synchronous output interface is responsible for writing data to the hardware component where the parameter _is the output message, WR is a control signal for writing and the writing operation is allowed after the flag } is reset. Notice that in the hardware component the *set* wire of the flag indicator is linked to the control wire l of the latch, which is driven by the control signal WR.

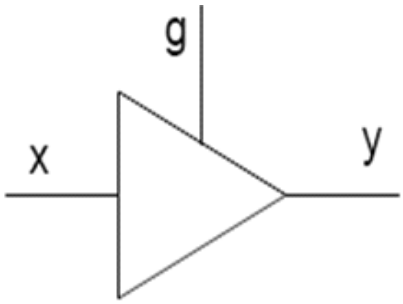

Figure 8(a) asynchronous interfaces

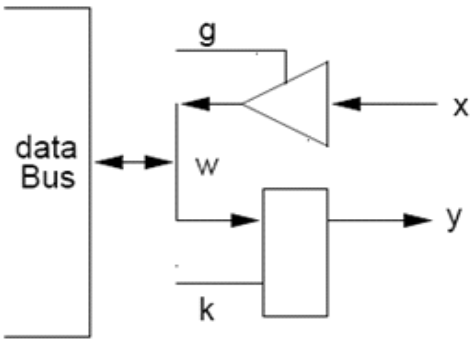

Figure 8(b) asynchronous interfaces

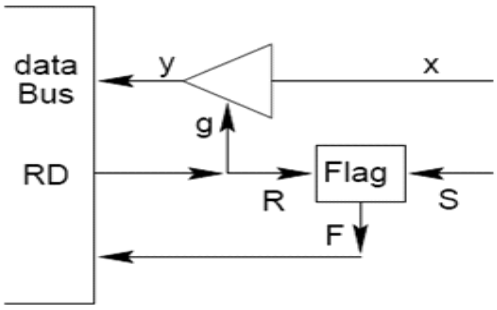

Figure 8(c) synchronous interfaces

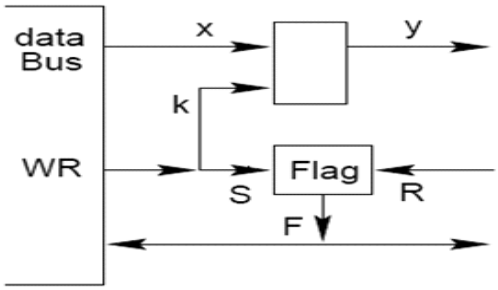

Figure 8(d) synchronous interfaces

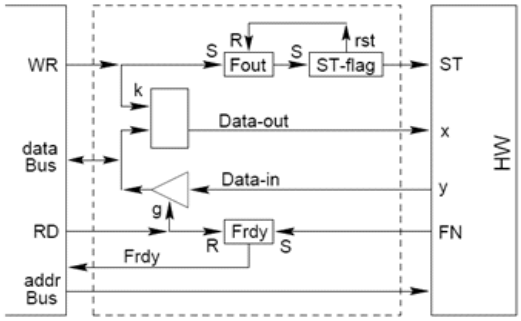

Figure 8(e) Hand shaking Protocol

Fig 8(e) shows typical hand shaking protocol to synchronise the hardware and software components. The *Data-out* wires from the software component connects to the input port '*x*' of the hardware component. The *Data-in* wires of the software component links to the output port '*y*' of the hardware component. The address bus links to the hardware directly. These are some of the common candidate *communication link architectures*. Many more hardware and software interfaces can be defined and implemented using bus extended technology.

VI.CONCLUSIONS

Sensor (actuator) fault model based on deterministic FSM is proposed and some types of observed faults are described. They are similar to the control flow and data flow faults. But this model can not represent inconsistencies in sensor readings [2]. Here sensor (actuator) integration testing with EPC is modeled as a diagnostic & fault localization method when the system specification and implementation are given in the form of sequence of communicating FSM (SCFSM). This method is described for the general case of diagnostics

DOI: 01.IJNS.02.02.79

problem and is applied to integration testing. With an example it is shown that the sensor (actuator) integration is a subset of fault diagnostic problem. From integration point of view, manifestation of faults through the interfaces is discussed.

At present manifestation of faults through the interfaces is under research for candidate architecture of mixed hardware/software, being used in a specific application.

REFERENCES

[1]. G. Heredia1, A. Ollero1, M. Bejar and R. Mahtani, Sensor and actuator fault detection in small autonomous helicopters (2008), Mechatronics, Vol. 18, N.2, pp. 90-99.

[2]. Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Greg Pottie, Mark Hansen, and Mani Srivastava "Sensor network data fault types", ACM Transactions on Sensor Networks (TOSN) Volume 5 , Issue 3 (May 2009) Article No.: 25

[3]. Edward Balaban, Abhinav Saxena, Prasun Bansal, Kai F. Goebel, Paul Stoelting and Simon Curran, "A Diagnostic Approach for Electro-Mechanical Actuators in Aerospace Systems" IEEEAC paper #1345, Version 4, Updated January 9, 2009

[4]. Bratthall, L.G., Runeson, P., Ädelsward, K., and Eriksson, W., A survey of lead-time challenges in the development and evolution of distributed real-time systems. Information and Software Technology, 42(13):947–958, September 2000.

[5]. Gerrit Muller, "Coping With System Integration Challenges in Large Complex Environments", Embedded Systems Institute.

[6]. Liangli, Ma; Houxiang, Wang; Li Yongjie, "A Reference Model of Grouped-Metadata Object and a Change Model based on it Appling for Component-based Software Integration Testing"- Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on 13-16 May 2007 Page(s):32 - 39

[7]. Seo, Kwang Ik; Choi, Eun Man; "Rigorous Vertical Software System Testing In IDE" Software Engineering Research, Management & Applications, 2007. SERA 2007. 5th ACIS International Conference on 20-22 Aug. 2007 Page(s):847 - 854

[8]. Bernhard K. Aichernig, Martin Weiglhofer, Franz Wotawa "Improving Fault-based Conformance Testing", Electronic Notes in Theoretical Computer Science 220 (2008) 63–77.

[9]. A.Petrenko, N. Yevtushenko, G. v. Bochmann, Fault models for testing in context, Proc. of the 1st Joint Intern. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing, and Verification. 1996, pp. 125-140.

[10]. Abdelaziz Guerrouat, and Harald Richter Adaptation of State/ Transition-Based Methods for Embedded System Testing, Proceedings Of World Academy Of Science, Engineering And Technology Volume 10 December 2005 ISSN 1307-688; p30-36.

[11]. Qiang Guo, Robert M. Hierons, Mark Harman and Karnig Derderian, "Heuristics for fault diagnosis when testing from finite state machines", Software Testing, Verification and Reliability Volume 17 Issue 1, Pages 41 – 57 Published Online**:** 22 Jun 2006.

[12]. Jens Chr. Godskesen. Connectivity Testing**,** Formal Methods in System Design Volume 25 , Issue 1 (July 2004 Pages: 5 - 38)

[13]. Ghedamsi and G. v. Bochmann. 'Test result analysis and diagnostics for finite state machines', Proc. of the 12-th ICDS, Yokohama, Japan, 1992.

[14]. Isermann, R. 2005. Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Springer.

[15]. Rolf Isermann, "Model-Based Fault Detection And Diagnosis - Status And Applications" Annual Reviews in Control 29 (2005) 71–85 – Elsevier.

[16]. T Amand, G Heyen, B Kalitventzeff, "Plant monitoring and fault detection:: Synergy between data reconciliation and principal component analysis" Computers & Chemical Engineering, 2001 – Elsevier

[17]. Patton, R., Uppal, F., Lopez-Toribio, C. (2000). Soft computing approaches to fault diagnosis for dynamic systems: a survey, 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes (Safeprocess'2000), Budapest, Hungary, Proceedings, pp. 298-311.

[18]. N. Aretakis, K. Mathioudakis and A. Stamatis, "Identification of sensor faults on turbofan engines using pattern recognition techniques" in Control Engineering Practice Volume12, Issue 7, July 2004, Pages 827-836;

[19]. S. Fujiwara, et al., "Test selection based on finite state models", IEEE transaction on Software Engineering 17(6): 591-603, 1991.

[20]. J. de. Kleer and B.c. Williams.'Diagnosing multiple faults', Artificial Intelligence 32(1), 1987, pp. 97-130.

[21]. Gill, Introduction to the Theory of Finite State Machines, McGraw-Hill, New York, 1962

[22]. Pieter Koopman, "Testing with Functions as Specifications", Dagstuhl Seminar Proceedings 04371 Perspectives of Model-Based Testing (2005). http://drops.dagstuhl.de/opus/volltexte/2005/324

[23]. J. A. Rowson and A. Sangiovanni-Vincentelli, "Interface-based design", *in Design Automation Conference*, pp. 178–183, June 1997.

[24]. D. Panigrahi, C. N. Taylor, and S. Dey, "Interface based hardware/software validation of a system-on-chip", *in High Level Design Validation and Test Workshop*, pp. 53–58, 2000.

[25] W. Ecker, V. Esen, T. Steininger, and M. Velten. HW/SW interface - implementation and modeling. In W. Ecker, W. M¨uller, and R. D¨omer, editors, *Hardware-dependent Software - Principlesand Practice*. Springer, 2008

DOI: 01.IJNS.02.02.79